BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

[redacted] [redacted]

Get this item

next ▶

PAJ 07-01-02 02202886 JP **APPLICATION DEVELOPMENT SYSTEM AND ITS METHOD AND APPLICATION DEVELOPMENT PROGRAM AND APPLICATION GENERATION METHOD**

INVENTOR(S)- SEKI, TAKEO

PATENT APPLICATION NUMBER- 2001323903
DATE FILED- 2001-10-22
PUBLICATION NUMBER- 02202886 JP
DOCUMENT TYPE- A
PUBLICATION DATE- 2002-07-19
PATENT PRIORITY INFO- 2000328819, 2000-10-27, Japan
INTERNATIONAL PATENT CLASS- G06F00944
APPLICANT(S)- TOSHIBA CORP
PUBLICATION COUNTRY- Japan NDN- 043-0254-1229-0

PROBLEM TO BE SOLVED: To provide an application development system capable of easily developing an application whose maintainability is excellent capable of flexibly facilitating countermeasures to the change of the system environment of a platform or the like. SOLUTION: A designing tool 1 supports the design of an application based on the combination of a plurality of logical components based on logical component information 4, and outputs logical design information 5 acquired from the design. A source generation processing part 2 and a compiler 3 generates an application (performable file 9) performable on a specific platform based on the logical design information 5 outputted by the designing tool 1 and the physical mounting information (physical component information 6 and component library 8) of the software components. COPYRIGHT: (C)2002,JPO

NO-DESCRIPTORS

---

back to top

Copyright Fee: **$ 0.00**
Document Price: **$ 13.50**
Total: $ 13.50*

add to cart

* If this item is not available from our original source at the price quoted you will be notified.
Shipping by US 1st Class Mail or eDoc is free of charge.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－202886
（P2002－202886A）
(43)公開日　平成14年７月19日(2002.7.19)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/44		Ｇ０６Ｆ　9/06	６２０Ａ　５Ｂ０７６

審査請求　未請求　請求項の数12　ＯＬ　（全 20 頁）

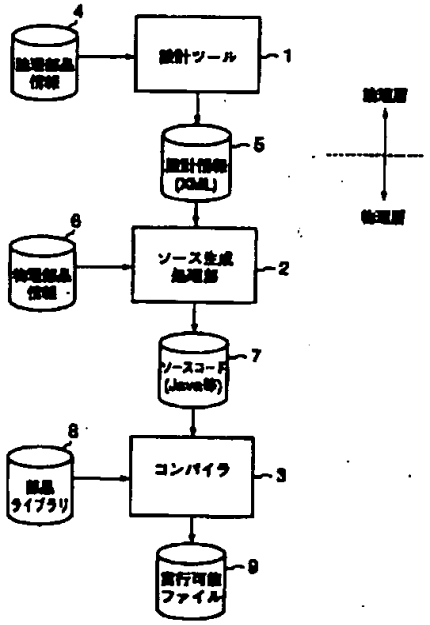
(21)出願番号　特願2001－323903(P2001－323903)
(22)出願日　平成13年10月22日(2001.10.22)
(31)優先権主張番号　特願2000－328819(P2000－328819)
(32)優先日　平成12年10月27日(2000.10.27)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(72)発明者　岡　武　夫
東京都府中市東芝町１番地　株式会社東芝
府中事業所内
(74)代理人　100075812
弁理士　吉武　賢次　（外６名）

Ｆターム(参考)　５Ｂ０７６ AB15 DA06 DB01 DB04 DD05

(54)【発明の名称】　アプリケーション開発システム、その方法およびアプリケーション開発プログラムおよびアプリケーション生成方法

(57)【要約】
【課題】　プラットフォーム等のシステム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる、アプリケーション開発システムを提供する。
【解決手段】　設計ツール１は、論理部品情報４に基づいて複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報５を出力する。ソース生成処理部２およびコンパイラ３は、設計ツール１により出力された論理的設計情報５とソフトウェア部品の物理的実装情報（物理部品情報６および部品ライブラリ８）とに基づいて特定のプラットフォーム上で実行可能なアプリケーション（実行可能ファイル９）を生成する。



【特許請求の範囲】
【請求項１】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発システムにおいて、
複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報を出力する論理設計部と、
前記論理設計部により出力された論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成する物理実装部とを備えたことを特徴とするアプリケーション開発システム。
【請求項２】前記論理設計部は、前記論理部品情報に基づいて前記各論理部品の仕様情報を生成する仕様情報生成部と、前記仕様情報生成部により生成された前記各論理部品の仕様情報に基づいて前記各論理部品の組み合わせによるアプリケーションの設計を支援する設計部本体とを有することを特徴とする請求項１記載のアプリケーション開発システム。
【請求項３】前記各論理部品の仕様情報は、前記各論理部品の入出力仕様と前記各論理部品間の接続仕様と前記各論理部品の属性仕様とを含むことを特徴とする請求項２記載のアプリケーション開発システム。
【請求項４】前記物理実装部は、前記論理設計部により出力された論理的設計情報と、システム環境向けに実装された物理部品と論理部品との間の対応関係を含む物理部品情報とに基づいて、ソースコードを生成するソース生成処理部と、前記ソース生成処理部により生成されたソースコードと前記各物理部品の実装情報が格納された部品ライブラリとに基づいてシステム環境上で実行可能な実行可能ファイルを生成する実行可能ファイル生成部とを有することを特徴とする請求項１記載のアプリケーション開発システム。
【請求項５】前記物理実装部は、システム環境向けに実装された物理部品と論理部品との間の対応関係を含む物理部品情報に基づいて、ソースコードを生成するソース生成処理部と、前記ソース生成処理部により生成されたソースコードと前記各物理部品の実装情報が格納された部品ライブラリとに基づいてシステム環境上で実行可能な実行可能ファイルを生成する実行可能ファイル生成部と、実行可能ファイル生成部により生成された実行可能ファイルの実行時に、前記論理設計部により出力された論理的設計情報と、前記物理部品情報とに基づいて、部品オブジェクトを生成して、部品の属性および部品間の接続関係を設定する部品オブジェクト生成処理部とを有することを特徴とする請求項１記載のアプリケーション開発システム。
【請求項６】前記論理的設計情報は、前記各論理部品の属性情報と前記各論理部品間の接続情報とを含むことを特徴とする請求項１乃至５のいずれか記載のアプリケーション開発システム。
【請求項７】前記論理的設計情報はＸＭＬ言語により記述されていることを特徴とする請求項１乃至６のいずれか記載のアプリケーション開発システム。
【請求項８】前記論理設計部および前記物理実装部はネットワークを介して互いに接続されたクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設けられ、前記クライアントコンピュータと前記サーバコンピュータとの間で前記論理的設計情報が受け渡されることを特徴とする請求項１乃至７のいずれか記載のアプリケーション開発システム。
【請求項９】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法において、
複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援するステップと、
前記設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成するステップとを含むことを特徴とするアプリケーション開発方法。
【請求項１０】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、
複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援させる手順と、
前記設計により得られた論理的設計情報を出力させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラム。
【請求項１１】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、
複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の組み合わせにより得られた論理的設計情報を読み取らせる手順と、
この読み取られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能

なアプリケーションを生成させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラム。

【請求項１２】複数のソフトウェア部品を組み合わせてアプリケーションを生成するアプリケーション生成方法において、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品の特定情報を表示し、ユーザーに対して、論理部品の選択を促すステップと、

ユーザーから選択された論理部品について、その定義情報を含む論理部品情報に基づいて論理的設計情報を生成するステップと、

前記論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてアプリケーションを生成するステップとを含むことを特徴とするアプリケーション生成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法に関する。

【０００２】

【従来の技術】従来から、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法が知られている。このような従来のアプリケーション開発方法においては、図１９に示すように、特定のプラットフォーム４１向けに実装されたソフトウェア部品である物理部品４２を物理的レベルで組み合わせることにより（符号４３参照）、アプリケーション４０の開発を行っている。

【０００３】

【発明が解決しようとする課題】しかしながら、上述した従来のアプリケーション開発方法では、ソフトウェア部品の組み合わせが物理的レベルで記述されることとなるので、ソフトウェア部品が実装されるプラットフォーム（ＯＳやミドルウェア、言語、通信等のシステム環境）に依存した記述が必要となり、開発されたアプリケーション４０を他のプラットフォームへ移行することが困難となるという問題がある。

【０００４】また、上述した従来のアプリケーション開発方法では、開発されるアプリケーション４０においてアプリケーション本来の処理（例えば業務処理）の記述とプラットフォーム固有の処理の記述とが混在することとなるので、アプリケーション４０の保守性が悪いという問題がある。

【０００５】本発明はこのような点を考慮してなされたものであり、プラットフォーム等のシステム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる、アプリケーション開発システム、その方法およびアプリケーション開発プログラムおよびアプリケーション生成方法を提供することを目的とする。

【０００６】

【課題を解決するための手段】図２０は本発明によるアプリケーション開発方法を模式的に説明するための図である。図２０に示すように、本発明では、複数のソフトウェア部品のそれぞれに対応して、プラットフォーム４１に依存しない部分を抽出してなる論理部品５２を設け、これら各論理部品５２を組み合わせることにより（符号５３参照）、アプリケーションの設計を行う。そして、このような設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報（物理部品４２と論理部品５２との間の対応関係を含む物理部品情報や、物理部品４２の実装情報が格納された部品ライブラリ）とに基づいて、特定のプラットフォーム４１上で実行可能なアプリケーション４０を生成する。

【０００７】このような基本原理の下で、本発明は、その第１の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発システムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報を出力する論理設計部と、前記論理設計部により出力された論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成する物理実装部とを備えたことを特徴とするアプリケーション開発システムを提供する。

【０００８】なお、上述した第１の解決手段においては、前記論理的設計情報はＸＭＬ言語により記述されていることが好ましい。

【０００９】また、上述した第１の解決手段においては、前記論理設計部および前記物理実装部はネットワークを介して互いに接続されたクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設けられ、前記クライアントコンピュータと前記サーバコンピュータとの間で前記論理的設計情報が受け渡されることが好ましい。

【００１０】また、本発明は、その第２の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法において、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援するステップと、前記設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシス

テム環境上で実行可能なアプリケーションを生成するステップとを含むことを特徴とするアプリケーション開発方法を提供する。

【0011】さらに、本発明は、その第3の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援させる手順と、前記設計により得られた論理的設計情報を出力させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラムを提供する。

【0012】さらにまた、本発明は、その第4の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の組み合わせにより得られた論理的設計情報を読み取らせる手順と、この読み取られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラムを提供する。

【0013】なお、本発明は、その第5の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションを生成するアプリケーション生成方法において、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品の特定情報を表示し、ユーザーに対して、論理部品の選択を促すステップと、ユーザーから選択された論理部品について、その定義情報を含む論理部品情報に基づいて論理的設計情報を生成するステップと、前記論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてアプリケーションを生成するステップとを含むことを特徴とするアプリケーション生成方法を提供する。

【0014】本発明によれば、ソフトウェア部品の組み合わせにより行われるアプリケーションの開発を、(1)ソフトウェア部品のうちプラットフォーム等のシステム環境に依存しない部分を抽出してなる論理部品を組み合わせてアプリケーションを設計する処理（論理層）と、(2)この設計により得られた論理的設計情報に基づいて、システム環境に依存したソフトウェア部品である物理部品を組み合わせてアプリケーションを生成する処理（物理層）とに分割して行うので、論理層の各部については、最終的なシステム環境にかかわらず共通に利用することが可能となり、物理層の各部を最終的なシステム環境に応じて個別に用意することにより、システム環境に対応したアプリケーションの開発を行うことができる。このため、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【0015】また、本発明によれば、論理層の処理と物理層の処理とを結び付ける論理的設計情報をXML言語という汎用性の高い言語で記述することにより、物理層および論理層の双方に自由度を与えることができる。このため、それぞれの層の差し替えを容易にし、しかも、アプリケーションの業務処理の論理表現の寿命を長くすることができる。

【0016】さらに、本発明によれば、論理層（論理設計部）および物理層（物理実装部）をクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設け、クライアントコンピュータとサーバコンピュータとの間で論理的設計情報を受け渡すようにすることにより、クライアントコンピュータとサーバコンピュータとの間での通信量を抑えることができる。このため、インターネット環境でも、クライアントコンピュータ側からサーバコンピュータ側のアプリケーションを開発することが可能となり、インターネット環境でのオープンなアプリケーション開発環境を提供するようなサービスに応用することが可能となる。

【0017】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態について説明する。図1乃至図11は本発明によるアプリケーション開発システムの一実施の形態を説明するための図である。なお、本実施の形態では、アプリケーション開発システムを単一のコンピュータ上で実現する場合を例に挙げて説明する。

【0018】図1に示すように、本実施の形態に係るアプリケーション開発システムは、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うものであり、設計ツール（論理設計部）1と、ソース生成処理部2およびコンパイラ（実行可能ファイル生成部）3とを備えている。なお、ソース生成処理部2およびコンパイラ3により物理実装部が構成されている。

【0019】このうち、設計ツール1は、論理部品情報4に基づいて複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報5を出力するものである。ここで、設計ツール1に入力される論理部品情報4は、論理部品の定義情報を含むものであり、例えばXML言語により文字列データとして記述されている（図7(a)(b)参照）。なお、論理部品とは、アプリケーションの設計のために用いられる部品であり、複数のソフトウェア部品のそれぞれに対応して設けられ、ソフトウェア部品のうちプラットフォーム（システム環境）に依存しない部分を抽出したものである。また、設計ツール1により出力される論

理的設計情報５は、各論理部品の属性情報と各論理部品間の接続情報とを含むものであり、例えばＸＭＬ言語により文字列データとして記述されている（図８参照）。

【００２０】一方、ソース生成処理部２およびコンパイラ３は、設計ツール１により出力された論理的設計情報５とソフトウェア部品の物理的実装情報（物理部品情報６および部品ライブラリ８）とに基づいて特定のプラットフォーム上で実行可能なアプリケーション（実行可能ファイル９）を生成するものである。

【００２１】具体的には、ソース生成処理部２は、設計ツール１により出力された論理的設計情報５と物理部品情報６とに基づいてソースコード７を生成するものである。なお、ソース生成処理部２に入力される物理部品情報６は、特定のプラットフォーム向けに実装された物理部品と論理部品との間の対応関係を含むものであり、例えばＸＭＬ言語により文字列データとして記述されている（図９および図１０(a)(b)(c)参照）。なお、論理部品と物理部品との対応関係は１対１に限らず、１対多でもよい。例えば、論理部品“Ａ”に対して物理部品“ａ”となる場合もあれば、論理部品“Ａ”に対して物理部品“ａ＋ｂ”となる場合もあり、同じ論理部品“Ａ”に対しても複数の物理部品を定義できる。さらに、より具体的には、論理部品“チェック付き更新”に対して物理部品が“UpdateWithCheck”となる場合もあれば、論理部品“チェック付き更新”に対して物理部品が“Check”＋“Update”となる場合もあり、同じ論理部品“チェック付き更新”に対しても複数の物理部品を定義できる。また、ソース生成処理部２により出力されるソースコード７は、例えばＪａｖａ（登録商標）等のプログラミング言語により文字列データとして記述されている（図１１参照）。

【００２２】また、コンパイラ３は、ソース生成処理部２により生成されたソースコード７と各物理部品の実装情報（インタフェースや型定義等）が格納された部品ライブラリ８とに基づいて特定のプラットフォーム上で実行可能な実行可能ファイル９を生成するものである。なお、部品ライブラリ８に格納される物理部品としては、ＥＪＢ、ＪａｖａＢｅａｎｓおよびＪａｖａクラス等が用いられる。

【００２３】以上において、設計ツール１、論理部品情報２および論理的設計情報５により論理層が構成され、ソース生成処理部２、コンパイラ３、物理部品情報６、ソースコード７、部品ライブラリ８および実行可能ファイル９により物理層が構成されている。

【００２４】図２は図１に示す設計ツール１の詳細を示す図である。

【００２５】図２に示すように、設計ツール１は、論理部品情報４に基づいて論理部品ツール情報１２を生成する論理部品ツール情報設定部（仕様情報生成部）１１と、論理部品ツール情報設定部１１により生成された論理部品ツール情報１２に基づいて各論理部品の組み合わせによるアプリケーションの設計を支援する設計ツール本体（設計部本体）１３とを有している。なお、論理部品情報４は、論理部品の定義情報として、論理部品基本情報４ａ（個々の論理部品のＩＤや属性情報、入出力情報等）と、論理部品関連情報４ｂ（個々の論理部品間の接続情報）とを含んでいる。また、論理部品ツール情報１２は、各論理部品の仕様情報を表すものであり、各論理部品の入出力仕様（論理部品入出力仕様１２ａ）と、各論理部品間の接続仕様（論理部品接続仕様１２ｂ）と、各論理部品の属性仕様（論理部品属性仕様１２ｃ）とを含んでいる。なお、設計ツール本体１３から出力される論理的設計情報５は、各論理部品の属性情報（部品属性情報５ａ）と、各論理部品間の接続情報（部品接続情報５ｂ）とを含んでいる。

【００２６】また、設計ツール１は、設計ツール本体１３に接続された画面入出力インタフェース部１４を有している。ここで、画面入出力インタフェース部１４は、設計の基礎となる複数の論理部品をツール画面上でアイコンとして表現するグラフィカルユーザインタフェース（ＧＵＩ）を提供するものであり、アイコン間を関係線で結んだグラフ構造により、開発対象となるアプリケーションを構成するソフトウェア部品の組み合わせを表現することができるようになっている。なお、画面入出力インタフェース部１４は、Ｗｅｂブラウザ等のアプリケーション上で動作するインタフェースプログラムとして実現することが可能である。

【００２７】図３はツール画面の一例を示す図である。図３に示すように、ツール画面２０は、パレット領域２１と、作業領域２２とを有している。このうち、パレット領域２１には、論理部品情報４に基づいて、設計の基礎となる論理部品に対応する複数のアイコン２３が一覧表示されている。また、作業領域２２には、パレット領域２１に配置されたアイコン２３に基づいて貼り付けられた複数のアイコン２４が表示されている。なお、作業領域２２に配置されたアイコン２４は、開発対象となるアプリケーションを構成する個々のソフトウェア部品に対応している。

【００２８】このようなツール画面２０において、ユーザは、マウス等の入力装置によりアイコンに対して各種の操作（アイコンの貼り付け、移動および関係付け等）を行うことができるようになっており、これらのアイコンに対する操作に連動して論理部品の組み合わせによりアプリケーションの設計を行うことができるようになっている。

【００２９】具体的には例えば、パレット領域２１に配置された複数のアイコン２３のうちの任意のアイコン（アイコンＡ，Ｂ，Ｃ）の上でマウスの左ボタンをクリックした後、作業領域２２の任意の位置でマウスの左ボタンを再度クリックすることにより、作業領域２２にア

イコン24（アイコンA，B，C）を貼り付けることができる（符号26参照）。なお、このようにして作業領域22にアイコン24が貼り付けられた後、アイコン24の上でマウスの右ボタンをクリックしてメニュー（図示せず）をポップアップし、このメニュー中の項目を適宜選択することにより、論理部品の属性等（論理部品の名前（"parent"，"child1"等）等）を設定することができる。

【0030】また、作業領域22に配置されたアイコンのうち、関係を設定する元のコンポーネントに対応するアイコン（図3では、アイコンA）の出力端子の上でマウスの左ボタンを押し、左ボタンを押した状態で、関係を設定する先のコンポーネントに対応するアイコン（図3では、アイコンB、C）までマウスを動かす（ドラッグする）ことにより、アイコン間の関係線25を定義することができる（符号27参照）。

【0031】なお、ツール画面20上でのアイコンに対する操作（アイコンの配置、移動および関係付け等）は、マウス等の基本操作により実現される。なお、マウス等の基本操作は、論理部品間で統一されている。

【0032】次に、このような構成からなる本実施の形態の作用について説明する。

【0033】まず、図4および図5により、図1および図2に示すアプリケーション開発システムの論理層（設計ツール1）の処理について説明する。

【0034】図4は図1および図2に示す設計ツール1の論理部品ツール情報設定部11の動作を説明するためのフローチャートである。

【0035】図4に示すように、設計ツール1の論理部品ツール情報設定部11は、まず、論理部品情報4のうち論理部品基本情報4aを読み取り（ステップ101）、この論理部品基本情報4aに基づいて、論理部品ごとに論理部品入出力仕様12aおよび論理部品属性仕様12cを生成する（ステップ102および103）。次に、論理部品情報4のうち論理部品関連情報4bを読み取り（ステップ104）、この論理部品関連情報4bに基づいて、論理部品の組み合わせごとに論理部品接続仕様12bを生成する（ステップ105）。

【0036】図7(a)(b)は論理部品情報4の一例を示す図である。このうち、図7(a)は、論理部品基本情報4aの一部（個々の論理部品のID）と論理部品関連情報4b（個々の論理部品間の接続情報）とを含む全体情報を示している。また、図7(b)は、論理部品基本情報4aの一部（個々の論理部品の属性情報および入出力情報）を含む個別情報を示している。

【0037】ここで、図7(a)の符号（A）の部分には、個々の論理部品のIDが記述されている（<;component type="A"/>;等）。また、符号（B）の部分には、論理部品接続仕様12bの元となる個々の論理部品間の接続情報が記述されている（<;connection from="A"to="B"/>;等）。なお、「<;connection from="A"to="B"/>;」の部分は、ソフトウェア部品Aからソフトウェア部品Bに接続できることを表している。

【0038】また、図7(b)の符号（C）の部分には、論理部品属性仕様12cの元となる個々の論理部品の属性情報が記述されている（<;property name="name" type="string"/>;等）。なお、「<;property name="name" type="string"/>; <;property name="option" type="string"/>;」の部分は、"name"と"option"という2つの文字列属性があることを表している。また、符号（D）の部分には、論理部品入出力仕様12aの元となる個々の論理部品の入出力情報が記述されている（<;terminalid="out" type="out" min=1 max=3>;...<;/terminal>;等）。なお、「<;terminal id="out" type="out" min=1 max=3>; <;connectwith type="B"/>; <;connectwith type="C"/>; <;/terminal>;」の部分は、"out"という出力端子があり、1本から3本の接続ができ、接続先はソフトウェア部品BまたはCであることを表している。

【0039】図5は図1および図2に示す設計ツール1の設計ツール本体13の動作を説明するためのフローチャートである。

【0040】図5に示すように、設計ツール1の画面入出力インタフェース部14は、設計ツール本体13による制御の下で、図3に示すツール画面20を呈示し、ユーザによるアイコンの操作を受け付けて、各論理部品の組み合わせによるアプリケーションの設計を支援する（ステップ201および202）。

【0041】具体的には、画面入出力インタフェース部14は、ユーザが、ツール画面20のパレット領域21に配置された複数のアイコン23のうちの任意のアイコン（アイコンA，B，C）の上でマウスの左ボタンをクリックした後、作業領域22の任意の位置でマウスの左ボタンを再度クリックすることにより、作業領域22にアイコン24（アイコンA，B，C）を貼り付ける（ステップ201）。

【0042】また、作業領域22に配置されたアイコンのうち、関係を設定する元のコンポーネントに対応するアイコン（図3では、アイコンA）の出力端子の上でマウスの左ボタンを押し、左ボタンを押した状態で、関係を設定する先のコンポーネントに対応するアイコン（図3では、アイコンB，C）までマウスを動かす（ドラッグする）ことにより、アイコン間の関係線25を定義する（ステップ202）。

【0043】その後、設計ツール本体13は、論理部品ツール情報12の論理部品入出力仕様12aに基づいて各論理部品の接続点の妥当性をチェックし（ステップ203）、妥当でない場合には、ステップ202に戻って処理を続け、妥当である場合には、ステップ205に進む（ステップ204）。

【0044】また、設計ツール本体13は、論理部品ツ

ール情報１２の論理部品接続仕様１２ｂに基づいて各論理部品間の接続関係を妥当性をチェックし（ステップ２０５）、妥当でない場合には、ステップ２０２に戻って処理を続け、妥当である場合には、ステップ２０７に進む（ステップ２０６）。

【００４５】その後、ユーザが、作業領域２２に貼り付けられたアイコン２４の上でマウスの右ボタンをクリックしてメニュー（図示せず）をポップアップし、このメニュー中の項目を適宜選択して論理部品の属性等を設定すると（ステップ２０７）、論理部品ツール情報１２の論理部品属性仕様１２ｃに基づいて各論理部品の属性の妥当性をチェックし（ステップ２０８）、妥当でない場合には、ステップ２０７に戻って処理を続け、妥当である場合には、ステップ２１０に進む（ステップ２０９）。

【００４６】最後に、全ての論理部品の処理が終了したか否かを判断し（ステップ２１０）、処理が終了している場合には、論理的設計情報５を出力する（ステップ２１１）。

【００４７】図８は論理的設計情報５の一例を示す図であり、論理的設計情報５は、部品属性情報５ａ（符号（Ｅ）参照）と部品接続情報５ｂ（符号（Ｆ）参照）とを含んでいる。ここで、図８に示す論理的設計情報５は、図３に示すような論理部品の組み合わせ（アイコンＡ（名前が"parent"で、オプションが"1"）の下にアイコンＢ（名前が"child1"で、オプションが"11"）とアイコンＣ（名前が"child2"で、オプションが"12"）とがツリー状に接続される組み合わせ）に対応している。

【００４８】次に、図６により、図１に示すアプリケーション開発システムの物理層（ソース生成処理部２およびコンパイラ３）の処理について説明する。

【００４９】図６は図１に示すソース生成処理部２の動作を説明するためのフローチャートである。

【００５０】図６に示すように、ソース生成処理部２は、設計ツール１により出力された論理的設計情報５（部品属性情報５ａおよび部品接続情報５ｂ）と物理部品情報６とに基づいてソースコード７を生成する。

【００５１】図９および図１０(a)(b)(c)は物理部品情報６の一例を示す図である。このうち、図９は、物理部品情報６のうちソフトウェア部品ごとのコード生成方法に関連する情報を示しており、特定のプラットフォーム向けに実装された物理部品と論理部品との間の対応関係や、個々の物理部品の属性関連情報および接続関連情報等を含んでいる。また、図１０(a)(b)(c)は、全体のコード生成方法に関連する情報を示している。

【００５２】ここで、図９の符号（Ｇ）の部分には、個々のソフトウェア部品の初期化コード生成方法が記述されている（<;new code="ClassX%name%=ClassX.newInstance()"/>;等）。また、符号（Ｈ）の部分には、個々のソフトウェア部品の属性設定コード生成方法が記述されている（<;property name="option" code="%name%.addItem("OPTION","%value%","string")"/>;等）。さらに、符号（Ｉ）の部分には、個々のソフトウェア部品の部品接続設定コード生成方法が記述されている（<;code="%from%.addItem("CHILD",%to%);/>;等）。なお、符号（Ｇ）（Ｈ）の部分および符号（Ｉ）の部分はそれぞれ、ソースコード７の属性設定コード７ｂおよび部品接続設定コード７ｃの元となる部分であり、「%xxx%」の項目が論理的設計情報５の値と置き換えられることによりソースコード７の該当部分が生成される。

【００５３】また、図１０(a)の符号（Ｊ）の部分には、ソースコード７の初期設定コード７ａの元となる初期設定コード生成方法が記述され（<;init file="fwlinit.java"/>;）、符号（Ｋ）の部分には、ソースコード７の終了処理設定コード７ｄの元となる終了処理設定コード生成方法が記述されている（<;term file="fwlterm.java"/>;）。なお、「<;init file="fwlinit.java"/>;」の部分は、図１０(b)に示すテンプレートファイル（fwlinit.java）をインポートするための記述であり、「<;term file="fwlterm.java"/>;」の部分は、図１０(c)に示すテンプレートファイル（fwlterm.java）をインポートするための記述である。

【００５４】このような物理部品情報６に基づいて、ソース生成処理部２は、図６に示すように、まず、ソースコード７の初期設定コード７ａを出力する（ステップ３０１）。

【００５５】次に、論理的設計情報５のうち部品属性情報５ａを読み取り（ステップ３０２）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとにソースコード７の属性設定コード７ｂを生成する（ステップ３０３）。

【００５６】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ３０４）、処理が終了していない場合には、ステップ３０２に戻って処理を続け、処理が終了している場合には、ステップ３０５に進む。

【００５７】次に、論理的設計情報５のうち部品接続情報５ｂを読み取り（ステップ３０５）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとにソースコード７の部品接続設定コード７ｃを生成する（ステップ３０６）。

【００５８】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ３０７）、処理が終了していない場合には、ステップ３０５に戻って処理を続け、処理が終了している場合には、物理部品情報６に基づいてソースコード７の終了処理設定コード７ｄを出力し（ステップ３０８）、全体の処理を終了する。図１１はこのようにして生成されるソースコード７の一例を示す図である。

【００５９】なお、このようにしてソース生成処理部２

によりソースコード７が生成された後、図１に示すように、ソースコード７をコンパイラ３にかけることにより、部品ライブラリ８に格納された各物理部品の実装情報に基づいて特定のプラットフォーム上で実行可能な実行可能ファイル９を生成する。

【００６０】このように本実施の形態によれば、ソフトウェア部品の組み合わせにより行われるアプリケーションの開発を、(1)ソフトウェア部品のうちプラットフォーム等のシステム環境に依存しない部分を抽出してなる論理部品を組み合わせてアプリケーションを設計する処理（論理層）と、(2)システム環境に依存したソフトウェア部品である物理部品を組み合わせてアプリケーションを生成する処理（物理層）とに分割して行うので、論理層の各部については、最終的なシステム環境にかかわらず共通に利用することが可能となり、物理層の各部を最終的なシステム環境に応じて個別に用意することにより、システム環境に対応したアプリケーションの開発を行うことができる。このため、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【００６１】具体的には、（１）論理層（設計ツール１）の処理により得られた論理的設計情報５に基づいて、複数のシステム環境に対応したアプリケーションを物理層の部品および処理（ソース生成処理部２、物理部品情報６、部品ライブラリ８およびコンパイラ３等）を差し替えることにより生成することができる。また、（２）論理層の処理により得られた論理的設計情報５は、システム環境に影響されることが少ないので、実行するシステム環境が技術進歩等により大きく変わった場合でも、新しいシステム環境に合わせた物理層の部品および処理をアプリケーションごとに用意することにより、新しいシステム環境上で実行可能なアプリケーションを容易に生成することができる。さらに、（３）論理層の処理は、物理層に縛られることなく特定の論理的設計情報５を出力することに限定することができるので、複数のシステム環境上で動作する開発ツールを容易に準備することができる。

【００６２】また、本実施の形態によれば、論理層の処理と物理層の処理とを結び付ける論理的設計情報５をＸＭＬ言語という汎用性の高い言語で記述しているので、物理層および論理層の双方に自由度を与えることができ、それぞれの層の差し替えを容易にし、しかも、アプリケーションの業務処理の論理表現の寿命を長くすることができる。

【００６３】なお、上述した実施の形態においては、ソース生成処理部２により、属性設定コードおよび部品接続設定コードを含むソースコード７を生成するとともに、コンパイラ３により、このソースコード７に基づいてソフトウェア部品の属性およびソフトウェア部品間の接続関係が設定された実行可能ファイル９を生成するようにしているが、図１２乃至図１７に示す他の実施の形態のように、ソフトウェア部品の属性およびソフトウェア部品間の接続関係については、実行時に設定するようにしてもよい。

【００６４】以下、図１２乃至図１７により、本発明によるアプリケーション開発システムの他の実施の形態について説明する。なお、図１２乃至図１７に示す実施の形態は、実行可能ファイルの実行時にソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定するようにした点を除いて、他は、図１乃至図１１に示す実施の形態と略同一である。図１２乃至図１７に示す実施の形態において、図１乃至図１１に示す実施の形態と同一部分には同一符号を付して詳細な説明は省略する。

【００６５】図１２に示すように、本発明の他の実施の形態においては、ソース生成処理部２′により、インタプリタ型部品オブジェクト生成コードを含むソースコード７を生成し、このソースコード７に基づいてコンパイラ３により生成された実行可能ファイル９の実行時に、実行処理部１５および部品オブジェクト生成処理部１６により、ソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定する。

【００６６】図１３は図１２に示すソース生成処理部２′の動作を説明するためのフローチャートである。

【００６７】図１３に示すように、ソース生成処理部２′は、物理部品情報６に基づいてソースコード７を生成する。

【００６８】図１５および図１６(a)(b)(c)(d)は物理部品情報６の一例を示す図である。このうち、図１５は、物理部品情報６のうちソフトウェア部品ごとの部品オブジェクト生成方法に関連する情報を示している。また、図１６(a)(b)(c)(d)は、全体のコード生成方法に関連する情報を示している。

【００６９】図１６(a)の符号（Ｌ）の部分には、ソースコード７の初期設定コード７ａ（図１３参照）の元となる初期設定コード生成方法が記述され（<;init file="fwlinit.java"/>;）、符号（Ｍ）の部分には、ソースコード７のインタプリタ型部品オブジェクト生成コード１７ｅ（図１３参照）の元となる部品オブジェクト生成方法が記述され（<;main file="fwlmain.java"/>;）、符号（Ｎ）の部分には、ソースコード７の終了処理設定コード７ｄ（図１３参照）の元となる終了処理設定コード生成方法が記述されている（<;term file="fwlterm.java"/>;）。なお、「<;init file="fwlinit.java"/>;」の部分は、図１６(b)に示すテンプレートファイル（fwlinit.java）をインポートするための記述であり、「<;main file="fwlmain.java"/>;」の部分は、図１６(c)に示すテンプレートファイル（fwlmain.java）をインポートするための記述であり、「<;term file="fwlterm.java"/>;」の部分は、図１６(d)に示すテンプレートファイル（fwlterm.java）をインポートするための記述である。

【0070】このような物理部品情報6に基づいて、ソース生成処理部2′は、図13に示すように、まず、ソースコード7の初期設定コード7aを出力する（ステップ401）。

【0071】次に、ソースコード7のインタプリタ型部品オブジェクト生成コード7eを出力する（ステップ402）。

【0072】その後、物理部品情報6に基づいてソースコード7の終了処理設定コード7dを出力し（ステップ403）、全体の処理を終了する。図17はこのようにして生成されるソースコード7の一例を示す図である。

【0073】なお、このようにしてソース生成処理部2′によりソースコード7が生成された後、図12に示すように、ソースコード7をコンパイラ3にかけることにより、部品ライブラリ8に格納された各物理部品の実装情報に基づいて特定のプラットフォーム上で実行可能な実行可能ファイル9を生成する。

【0074】このようにして生成された実行可能ファイル9は実行処理部15により実行されるが、実行処理部15はその処理の過程で部品オブジェクト生成処理部16を呼び出し、これにより、ソフトウェア部品の部品オブジェクトを生成するとともに、当該ソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定する。

【0075】図14は図12に示す部品オブジェクト生成処理部16の動作を説明するためのフローチャートである。

【0076】図14に示すように、部品オブジェクト生成処理部16は、設計ツール1により出力された論理的設計情報5（部品属性情報5aおよび部品接続情報5b）と物理部品情報6とに基づいて、部品オブジェクトの生成、および属性および部品間の接続情報の設定等を行う。

【0077】具体的には、まず、論理的設計情報5のうち部品属性情報5aを読み取り（ステップ501）、この部品属性情報5aと物理部品情報6とに基づいて、ソフトウェア部品ごとに部品オブジェクトを生成し、属性を設定する（ステップ502）。

【0078】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ503）、処理が終了していない場合には、ステップ501に戻って処理を続け、処理が終了している場合には、ステップ504に進む。

【0079】次に、論理的設計情報5のうち部品接続情報5bを読み取り（ステップ504）、この部品属性情報5aと物理部品情報6とに基づいて、ソフトウェア部品ごとに相互参照情報を設定し、部品間を接続する（ステップ505）。

【0080】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ506）、処理が終了していない場合には、ステップ504に戻って処理を続け、処理が終了している場合には、全体の処理を終了する。

【0081】なお、上述した図1乃至図11に示す実施の形態、および図12乃至図17に示す実施の形態においては、アプリケーション開発システムを単一のコンピュータ上で実現する場合を例に挙げて説明したが、これに限らず、アプリケーション開発システムを図18に示すような分散開発環境で実現することも可能である。具体的には、図18に示すように、インターネット等のネットワーク32を介して互いに接続された複数のクライアントコンピュータ31およびサーバコンピュータ33において、論理層（設計ツール1および論理部品情報4）をクライアントコンピュータ31上に設け、物理層（ソース生成処理部2、コンパイラ3、物理部品情報6、ソースコード7、部品ライブラリ8および実行可能ファイル9）をサーバコンピュータ33上に設けるようにしてもよい。なお、この場合には、クライアントコンピュータ31とサーバコンピュータ33との間で、物理部品の実装情報を伴わないXML言語等による論理的設計情報5が受け渡されることとなるので、通信量を抑えることができる。このため、インターネット環境でも、クライアントコンピュータ31側からサーバコンピュータ33側のアプリケーションを開発することが可能となり、インターネット環境でのオープンなアプリケーション開発環境を提供するようなサービスに応用することが可能となる。

【0082】また、上述した図1乃至図11に示す実施の形態、および図12乃至図17に示す実施の形態においては、コンパイラ3により、ソース生成処理部2により生成されたソースコード7と部品ライブラリ8とを統合しているが、これに限らず、ソースコード7と部品ライブラリ8とを部品選択手法や部品プラグイン手法等により統合するようにしてもよい。

【0083】さらに、上述した図1乃至図11に示す実施の形態、および図12乃至図17に示す実施の形態においては、論理部品情報4および物理部品情報6をXML言語により記述しているが、これに限らず、表形式等の任意の形式で記述することができる。

【0084】なお、上述した図1乃至図11に示す実施の形態、および図12乃至図17に示す実施の形態において、アプリケーション開発システムの設計ツール1、ソース生成処理部2およびコンパイラ3はプログラムとして実現することができる。このようなプログラムからなるアプリケーション開発プログラムは、例えば図18に示すような分散開発環境であれば、フレキシブルディスク34やCD-ROM35等のコンピュータ読み取り可能な記録媒体に記録することが可能であり、クライアントコンピュータ31およびサーバコンピュータ33上において、図4乃至図6に示すような手順および図13

および図14に示すような手順に従ってアプリケーションを開発することができる。

【0085】なお、本発明における記録媒体としては、フレキシブルディスクやCD-ROMに限られず、磁気ディスク、内部メモリ、ハードディスク、光ディスク（CD-R、DVD（Digital Versatile Disk）等）、光磁気ディスク（MO（Magneto Optical）等）、半導体メモリ等のように、プログラムを記録でき、かつコンピュータが読み取り可能な記録媒体であれば、その記録形式はいずれかの形式であってもよい。また、記録媒体としては、ネットワーク上で伝送される際の搬送波や、情報伝達媒体も含む。

【0086】また、記録媒体からコンピュータにインストールされたプログラムの指示に基づきコンピュータ上で稼働しているOS（オペレーティングシステム）や、データベース管理ソフト、ネットワークソフト等のMW（ミドルウェア）等が本発明を実現するための各処理の一部を実行してもよい。

【0087】さらに、本発明における記録媒体は、コンピュータと独立した媒体に限らず、LANやインターネット等により伝送されたプログラムをダウンロードして記憶または一時記憶した記録媒体も含まれる。

【0088】さらにまた、記録媒体は1つに限らず、複数の媒体から本発明における処理が実行される場合も本発明における記録媒体に含まれ、媒体構成はいずれの構成であってもよい。

【0089】

【発明の効果】以上説明したように本発明によれば、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【図面の簡単な説明】

【図1】本発明によるアプリケーション開発システムの一実施の形態を示すシステム構成図。

【図2】図1に示すアプリケーション開発システムの設計ツール（論理設計部）の詳細を示すブロック図。

【図3】図1に示すアプリケーション開発システムにおいてユーザに対して呈示されるツール画面の一例を示す図。

【図4】図1および図2に示す設計ツールの論理部品ツール情報設定部（仕様情報生成部）の動作を説明するためのフローチャート。

【図5】図1および図2に示す設計ツールの設計ツール本体（設計部本体）の動作を説明するためのフローチャート。

【図6】図1に示すソース生成処理部の動作を説明するためのフローチャート。

【図7】図1に示すアプリケーション開発システムで用いられる論理部品情報の一例を示す図。

【図8】図1に示すアプリケーション開発システムで出力される論理的設計情報の一例を示す図。

【図9】図1に示すアプリケーション開発システムで用いられる物理部品情報（ソフトウェア部品ごとのコード生成方法に関連する情報）の一例を示す図。

【図10】図1に示すアプリケーション開発システムで用いられる物理部品情報（全体のコード生成方法に関連する情報）の一例を示す図。

【図11】図1に示すアプリケーション開発システムで生成されるソースコードの一例を示す図。

【図12】本発明によるアプリケーション開発システムの他の実施の形態を示すシステム構成図。

【図13】図12に示すソース生成処理部の動作を説明するためのフローチャート。

【図14】図12に示す部品オブジェクト生成処理部の動作を説明するためのフローチャート。

【図15】図12に示すアプリケーション開発システムで用いられる物理部品情報（ソフトウェア部品ごとのコード生成方法に関連する情報）の一例を示す図。

【図16】図12に示すアプリケーション開発システムで用いられる物理部品情報（全体のコード生成方法に関連する情報）の一例を示す図。

【図17】図12に示すアプリケーション開発システムで生成されるソースコードの一例を示す図。

【図18】本発明によるアプリケーション開発システムが適用される分散開発環境の一例を示す図。

【図19】従来のアプリケーション開発方法を模式的に説明するための図。

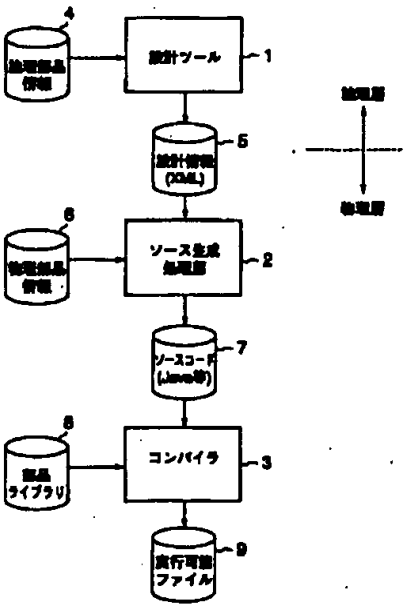
【図20】本発明によるアプリケーション開発方法を模式的に説明するための図。

【符号の説明】

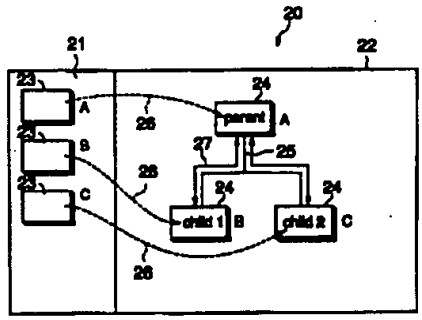
1　設計ツール（論理設計部）
2，2'　ソース生成処理部（物理実装部）
3　コンパイラ（物理実装部）
4　論理部品情報
4a　論理部品基本情報
4b　論理部品関連情報
5　論理的設計情報
5a　部品属性情報
5b　部品接続情報
6　物理部品情報
7　ソースコード
7a　初期設定コード
7b　属性設定コード
7c　部品接続設定コード
7d　終了処理設定コード
8　部品ライブラリ
9　実行可能ファイル
11　論理部品ツール情報設定部（仕様情報生成部）
12　論理部品ツール情報（各論理部品の仕様情報）
12a　論理部品入出力仕様
12b　論理部品接続仕様

１２ｃ　論理部品属性仕様
１３　設計ツール本体（設計部本体）
１４　画面入出力インタフェース部
１５　実行処理部
１６　部品オブジェクト生成処理部
２０　ツール画面
２１　パレット領域
２２　作業領域
２３，２４　アイコン
２５　関係線
３１　クライアントコンピュータ
３２　ネットワーク
３３　サーバコンピュータ
３４　フレキシブルディスク（記録媒体）
３５　ＣＤ－ＲＯＭ（記録媒体）
４０　アプリケーション
４１　プラットフォーム（システム環境）
４２　物理部品
４３　物理的な組み合わせ情報
５２　論理部品
５３　論理的な組み合わせ情報

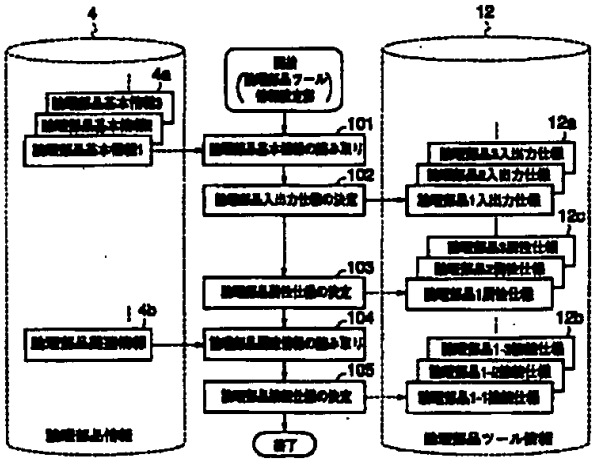
【図１】



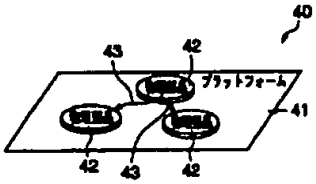
【図３】



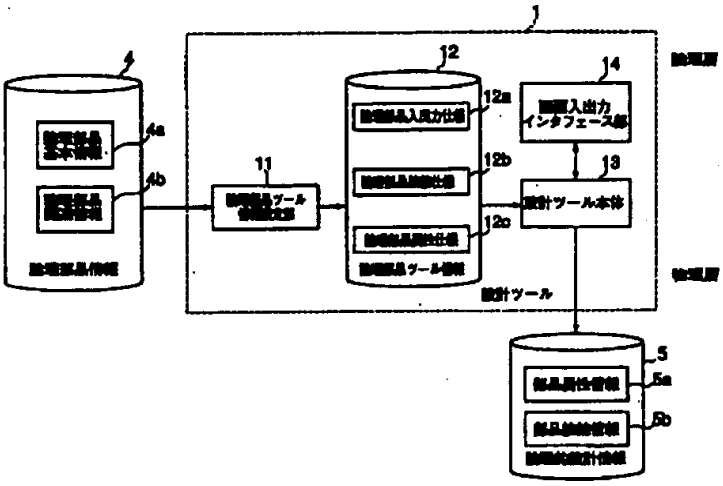
【図４】



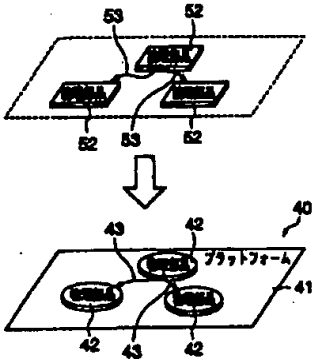
【図１９】



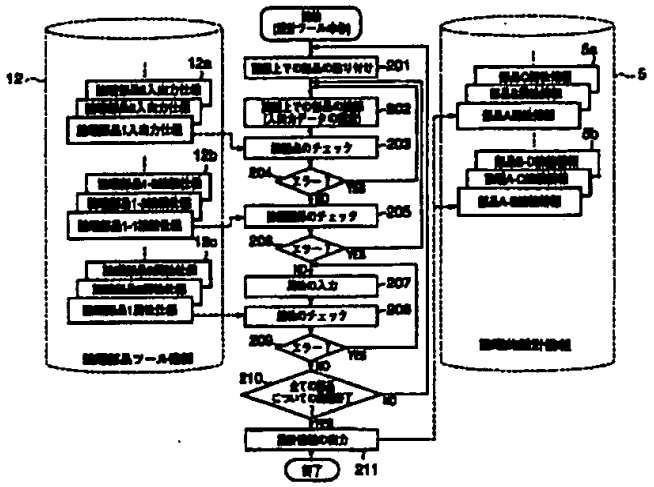
【図2】


1
4
4a
4b
11
12
12a
12b
12c
13
14
設計ツール
5
5a
5b

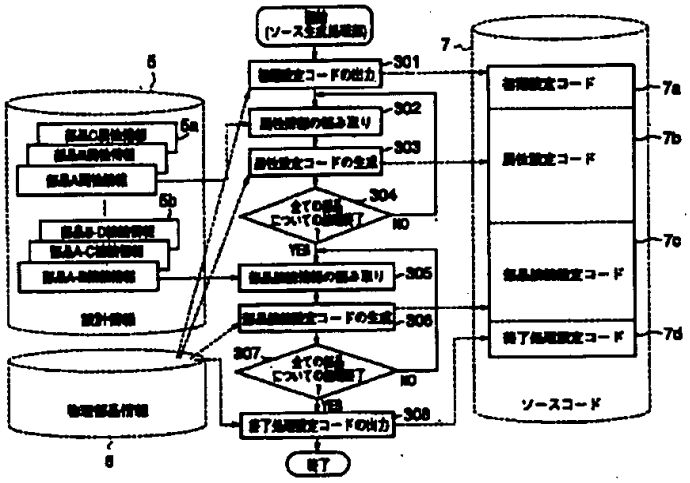
【図20】


53
52
52
53
52
40
43
42
プラットフォーム
41
42
43
42

【図5】


12
12a
12b
12c
201
202
203
204
エラー
205
206
エラー
207
208
エラー
209
210
211
5
5a
5b

【図6】



【図8】

```
<component type="A",name="parent" option=1>
</component>
<component type="B",name="child1" option=11>
</component>
<component type="C",name="child2" option=12>
</component>
<connection from="parent" to="child1" />
<connection from="parent" to="child2" />
```

Ⓔ Ⓕ

【図15】

```
<componentimpl name="A" impleclass="ClassX">
        <property name="option" type=string"/>
</componenttype>
<componentimpl name="B" impleclass="classY">
        <property name="option" type=string"/>
</componenttype>
<componentimpl name="C" impleclass="classZ">
        <property name="option" type=string"/>
</componenttype>
```

【図7】

(a)

```
<framework name="fw1">
        <component type="A"/>  ···Ⓐ
        <component type="B"/>  ···Ⓐ
        <component type="C"/>  ···Ⓐ
        <connection from="A" to="B"/>
        <connection from="A" to="C"/>      } Ⓑ
</framework>
```

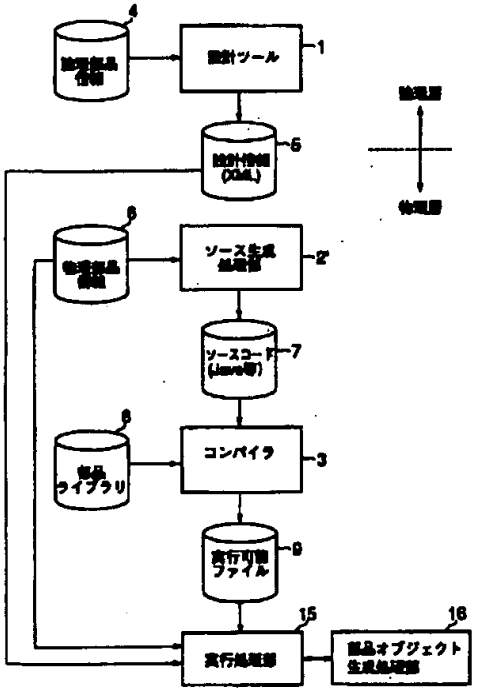
(b)

```
<componenttype name="A">
        <property name="name"type="string" />          } Ⓒ
        <property name="option"type="string" />
        <terminal id="out" type="out" min=1 max=3>     } Ⓓ
                <connectwith type="B"/>
                <connectwith type="C"/>
        </terminal>
</componenttype>
<componenttype name="B">
        <property name="name"type="string" />
        <property name="option"type="string" />
        <terminal id="in" type="in" min=1 max=1>
                <connectwith type="A"/>
        </terminal>
</componenttype>
<componenttype name="C">
        <property name="name"type="string" />
        <property name="option"type="string" />
        <terminal id="in" type="in" min=1 max=1>
                <connectwith type="A"/>
        </terminal>
</componenttype>
```

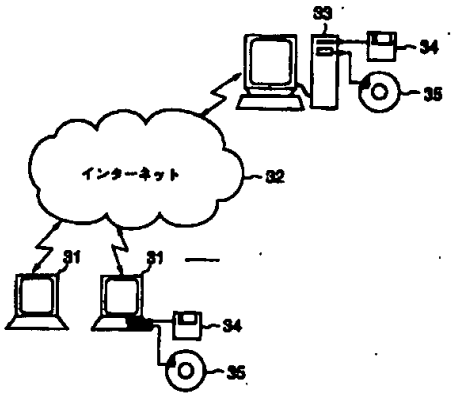
【図9】

```
<componentimpl name="A" implclass="ClassX">
        <new code="ClassX%name%=ClassX.newinstance()" />                    ⑥
        <property name="option"code="%name%.addItem("OPTION","%value%","string")" />  ⑧
</componenttype>
<componentimpl name="B" implclass="classY">
        <new code="ClassY%name%=ClassY.newinstance()" />
        <property name="option"code="%name%.addItem("OPTION","%value%","string")" />
</componenttype>
<componentimpl name="C" implclass="classZ">
        <new code="ClassZ%name%=ClassZ.newinstance()" />
        <property name="option"code="%name%.addItem("OPTION","%value%","string")" />
</componenttype>
<connectionimpls>
        <code="%from%.addItem("CHILD",%to%); />
        <code="%to%.addItem("PARENT",%from%); />                            } ①
</componenttype>
```

【図12】



【図18】



【図10】

(a)

```
<frameworkimpl name="fw1" >
        <init file="fw1init.java" /> --------- ㋚
        <term file="fw1term.java" /> ------- ㋛
</componenttype>
```

fw1 init.java

(b)

```
import java. lang. *;
import java. util. *;
    •
    • (各種定義項目)
    •

public class AA1 implements AA
{
        public Object bbbb()
        {
          •
          • (各種初期化処理)
          •
```

fw1 term.java

(c)

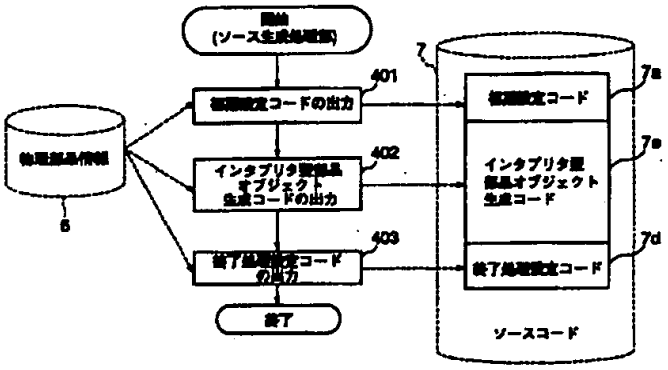
```
          •
          • (各種終了処理)
          •
         )
}
```

【図11】

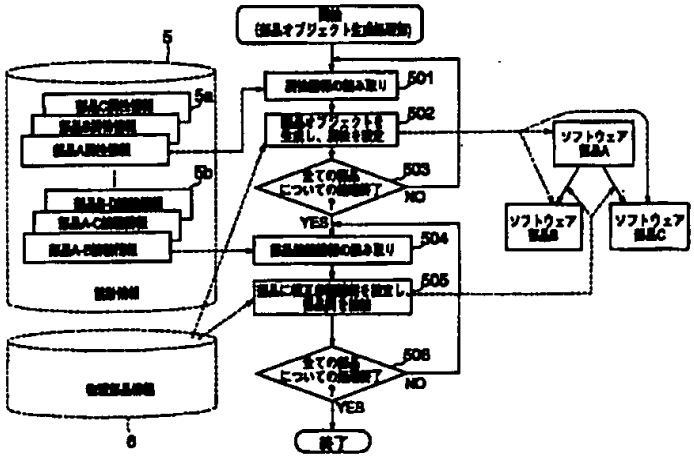
```
import java. lang. *;
import java. util. *;
    ・
    ・(各種定義項目)
    ・

public class AA1 implements AA
{
        public Object bbbb()
        {
          ・
          ・(各種初期化処理)
          ・
        ClassX parent=ClassX.newinstance();
        parent.addItem("OPTION","1","string");
        ClassY child1=ClassY.newinstance();
        child1.addItem("OPTION","11","string");
        ClassZ parent=ClassZ.newinstance();
        parent.addItem("OPTION","12","string");
        parent.addItem("CHILD",child1);
        child1.addItem("PARENT",parent);
        parent.addItem("CHILD",child2);
        child2.addItem("PARENT",parent);
          ・
          ・(各種終了処理)
          ・
        }
}
```

【図13】


開始
(ソース生成処理部)
401
402
403
終了
6
7
7a
7b
7d
インタプリタ部
部品オブジェクト
生成コード
ソースコード

【図14】


開始
(部品オブジェクト生成処理部)
5
5a
5b
501
502
503
NO
YES
504
505
506
NO
YES
終了
6
ソフトウェア
部品A
ソフトウェア
部品B
ソフトウェア
部品C

【図16】

(a)

```
<frameworkimpl name="fw1">
        <init file="fw1init.java"/> -------- (L)
        <main file="fw1main.java"/>---- (M)
        <term file="fw1term.java"/>------(N)
</componenttype>
```

fw1 init.java

(b)

```
import java. lang. *;
import java. util. *;
    •
    •(各種定義項目)
    •

public class AA1 implements AA
{
        public Object bbbb()
        {
          •
          •(各種初期化処理)
          •
```

fw1main.java

(c)

```
          •
          •
          •
          •(インタプリタ型部品オブジェクト生成処理)
          •
          •
          •
```

fw1term.java

(d)

```
          •
          •(各種終了処理)
          •
        }
}
```

【図17】

```
import java. lang. *;
import java. util. *;
    •
    • (各種定義項目)
    •

public class AA1 implements AA
{
        public Object bbbb()
        {
          •
          • (各種初期化処理)
          •
          •
          •
          • (インタプリタ型製品オブジェクト
          •                    生成処理)
          •
          •
          •
          •
          •
          • (各種終了処理)
          •
        }
}
```

PATENT ABSTRACTS OF JAPAN

(11)Publication number : **2002-202886**
(43)Date of publication of application : **19.07.2002**

(51)Int.CI. **G06F 9/44**

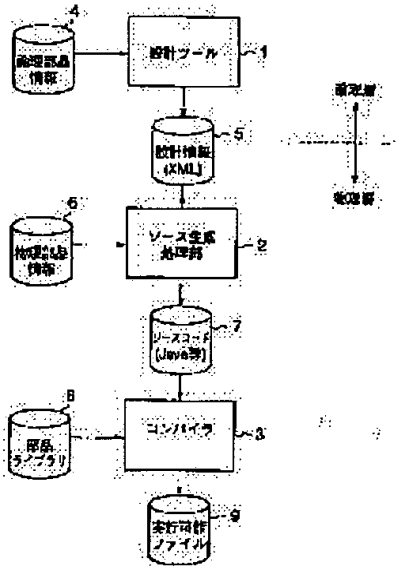
(21)Application number : **2001-323903**
(22)Date of filing : **22.10.2001**

(71)Applicant : **TOSHIBA CORP**
(72)Inventor : **SEKI TAKEO**

(30)Priority
Priority number : **2000328819** Priority date : **27.10.2000** Priority country : **JP**

(54) APPLICATION DEVELOPMENT SYSTEM AND ITS METHOD AND APPLICATION DEVELOPMENT PROGRAM AND APPLICATION GENERATION METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an application development system capable of easily developing an application whose maintainability is excellent capable of flexibly facilitating countermeasures to the change of the system environment of a platform or the like.
SOLUTION: A designing tool 1 supports the design of an application based on the combination of a plurality of logical components based on logical component information 4, and outputs logical design information 5 acquired from the design. A source generation processing part 2 and a compiler 3 generates an application (performable file 9) performable on a specific platform based on the logical design information 5 outputted by the designing tool 1 and the physical mounting information (physical component information 6 and component library 8) of the software components.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] this invention relates to the application development method of developing application combining two or more software parts.
[0002]
[Description of the Prior Art] From the former, the application development method of developing application combining two or more software parts is learned. In such a conventional application development method, as shown in drawing 19 , (sign 43 reference) and development of application 40 are performed by combining the specific physical parts 42 which are software parts mounted for turning plat-form 41 on physical level.
[0003]
[Problem(s) to be Solved by the Invention] However, since the combination of software parts will be described by physical level by the conventional application development method mentioned above, description depending on the plat form (system environment, such as OS, middleware, a language, and communication) where software parts are mounted is needed, and there is a problem that it becomes difficult to shift the developed application 40 to other plat forms.
[0004] Moreover, by the conventional application development method mentioned above, since description of processing (for example, operating processing) of application original and description of processing peculiar to a plat form will be intermingled in the application 40 developed, there is a problem that the maintainability of application 40 is bad.
[0005] this invention is made in consideration of such a point, and it aims at offering the application development system which can develop easily application excellent in the maintainability which can respond to change of the system environment of a plat form etc. flexibly, its method, an application development program, and an application generation method.
[0006]
[Means for Solving the Problem] Drawing 20 is drawing for explaining the application development method by this invention typically. As shown in drawing 20 , in this invention, corresponding to each of two or more software parts, the Boolean part article 52 which comes to extract the portion independent of a plat form 41 is formed, and (sign 53 reference) and the design of application are performed by combining each [these] Boolean part article 52. And based on the logical design information and the physical mounting information on software parts (part library where physical part information including the correspondence relation between the physical parts 42 and the Boolean part article 52 and the mounting information on the physical parts 42 were stored) which were acquired by such design, the application 40 which can be performed on the specific plat form 41 is generated.
[0007] In the application development system to which this invention develops application as the 1st solution means under such a basic principle combining two or more software parts It is based on the Boolean part article information including the definition information on the Boolean part article that it comes to extract the portion for which is the Boolean part article prepared corresponding to each of two or more software parts, and it does not depend on system environment among the software parts concerned. The logical design section which outputs the logical design information which supported the design of the application by the combination of two or more Boolean part articles, and was acquired by the design concerned, The application development system characterized by having the physical mounting section which generates the application which can be performed on system environment based on the logical design information and the physical mounting information on software parts which were outputted by the aforementioned logical design section is offered.
[0008] In addition, as for the aforementioned logical design information, in the 1st solution means mentioned above, being described by the XML language is desirable.
[0009] Moreover, as for the aforementioned logical design section and the aforementioned physical mounting section, in the 1st solution means mentioned above, it is desirable that it is prepared, respectively on the client computer each other connected through the network and a server computer, and the aforementioned logical design information is delivered between the aforementioned client computer and the aforementioned server computer.
[0010] Moreover, this invention is set to the application development method of developing application as the 2nd solution means combining two or more software parts. It is based on the Boolean part article information including the definition information on the Boolean part article that it comes to extract the portion for which is the Boolean part article prepared corresponding to each of two or more software parts, and it does not depend on system environment among the software parts concerned. The step which supports the design of the application by the combination of two or more Boolean part articles, The application development method characterized by including the step which generates the application which can be performed on system environment based

on the logical design information and the physical mounting information on software parts which were acquired by the aforementioned design is offered.

[0011] Furthermore, this invention is set to the application development program which develops application as the 3rd solution means combining two or more software parts. It is based on the Boolean part article information including the definition information on the Boolean part article that it comes to extract the portion for which is the Boolean part article prepared corresponding to each of two or more software parts, and it does not depend on system environment among the software parts concerned. The application development program characterized by performing the procedure of making the design of the application by the combination of two or more Boolean part articles supporting, and the procedure to which the logical design information acquired by the aforementioned design is made to output to a computer is offered.

[0012] In the application development program to which this invention develops application as the 4th solution means further again combining two or more software parts The procedure of making the logical design information acquired with the combination of the Boolean part article which comes to extract the portion for which is the Boolean part article prepared corresponding to each of two or more software parts, and it does not depend on system environment among the software parts concerned reading, The application development program characterized by performing the procedure of making the application which can be performed on system environment generating based on this logical design information and physical mounting information on software parts that were read, to a computer is offered.

[0013] In addition, this invention is set to the application generation method which generates application as the 5th solution means combining two or more software parts. Display the specific information of the Boolean part article prepared corresponding to each of two or more software parts, and a user is received. The step to which selection of the Boolean part article is urged, and the step which generates logical design information about the Boolean part article chosen from the user based on the Boolean part article information including the definition information, The application generation method characterized by including the step which generates application based on the aforementioned logical design information and the physical mounting information on software parts is offered.

[0014] According to this invention, the development of application performed with the combination of software parts (1) Processing which designs application combining the Boolean part article which comes to extract the portion for which it does not depend on the system environment of a plat form etc. among software parts (logic layer), (2) Since it carries out by dividing into the processing (physical layer) which generates application combining the physical parts which are software parts depending on system environment based on the logical design information acquired by this design About each part of a logic layer, irrespective of final system environment, it can become possible to use in common and application corresponding to system environment can be developed by preparing each part of a physical layer individually according to final system environment. For this reason, application excellent in the maintainability which can respond to change of system environment flexibly can be developed easily.

[0015] Moreover, according to this invention, flexibility can be given to the both sides of a physical layer and a logic layer by describing the logical design information which connects processing of a logic layer, and processing of a physical layer in the high language of the versatility of an XML language. For this reason, substitution of each layer can be made easy and, moreover, the life of the logical expression of operating processing of application can be lengthened.

[0016] Furthermore, according to this invention, the traffic between a client computer and a server computer can be stopped by preparing a logic layer (logical design section) and a physical layer (physical mounting section) on a client computer and a server computer, respectively, and delivering logical design information between a client computer and a server computer. For this reason, it becomes possible also in the Internet environment to develop the application by the side of [a client computer side to] a server computer, and it becomes possible to apply to service which offers the open application development environment in the Internet environment.

[0017]

[Embodiments of the Invention] Hereafter, the gestalt of operation of this invention is explained with reference to a drawing. Drawing 1 or drawing 11 is drawing for explaining the gestalt of 1 operation of the application development system by this invention. In addition, the case where application development system is realized on a single computer is mentioned as an example, and the gestalt of this operation explains it.

[0018] As shown in drawing 1 , the application development system concerning the gestalt of this operation develops application combining two or more software parts, and is equipped with the design tool (logical design section) 1, and the source generation processing section 2 and a compiler (executable-file generation section) 3. In addition, the physical mounting section is constituted by the source generation processing section 2 and the compiler 3.

[0019] Among these, the design tool 1 supports the design of the application by the combination of two or more Boolean part articles based on the Boolean part article information 4, and outputs the logical design information 5 acquired by the design concerned. Here, the Boolean part article information 4 that it is inputted into the design tool 1 is described as character-string data by the XML language including the definition information on the Boolean part article (refer to drawing 7 (a) and (b)). In addition, the Boolean part articles are parts used for the design of application, it is prepared corresponding to each of two or more software parts, and the portion for which it does not depend on a plat form (system environment) among software parts is extracted. Moreover, the logical design information 5 outputted by the design tool 1 is described as character-string data by the XML language including the attribute information on each Boolean part article, and the initial entry between each Boolean part article (refer to drawing 8).

[0020] On the other hand, the source generation processing section 2 and a compiler 3 generate the application (executable file 9) which can be performed on a specific plat form based on the logical design information 5 and the physical mounting information

on software parts (the physical part information 6 and part library 8) which were outputted by the design tool 1.
[0021] Specifically, the source generation processing section 2 generates a source code 7 based on the logical design information 5 and the physical part information 6 which were outputted by the design tool 1. In addition, the physical part information 6 that it is inputted into the source generation processing section 2 is described as character-string data by the XML language including the correspondence relation between the physical parts and the Boolean part articles which were mounted for [specific] plat forms (refer to drawing 9 and drawing 10 (a), (b), and (c)). In addition, not only 1 to 1 but one-pair ** is sufficient as the correspondence relation between the Boolean part article and physical parts. For example, if it may become physical parts "a" to the Boolean part article "A", it may become physical parts "a+b" to the Boolean part article "A", and two or more physical parts can be defined also to the same Boolean part article "A." Furthermore, more specifically, if physical parts may serve as "UpdateWithCheck" to the Boolean part article "updating with a check", physical parts may become "Check"+ "Update" to the Boolean part article "updating with a check", and two or more physical parts can be defined also to the same Boolean part article "updating with a check." Moreover, the source code 7 outputted by the source generation processing section 2 is described as character-string data by programming language, such as Java (registered trademark), (refer to drawing 11).
[0022] Moreover, a compiler 3 generates the executable file 9 which can be performed on a specific plat form based on the source code 7 generated by the source generation processing section 2 and the part library 8 where the mounting information on each physical part (an interface, type definition, etc.) was stored. In addition, as physical parts stored in the part library 8, EJB, Java Beans, a Java class, etc. are used.
[0023] A logic layer is constituted by the design tool 1, the Boolean part article information 2, and the logical design information 5 above, and the physical layer is constituted by the source generation processing section 2, a compiler 3, the physical part information 6, the source code 7, the part library 8, and the executable file 9.
[0024] Drawing 2 is drawing showing the detail of the design tool 1 shown in drawing 1 .
[0025] As shown in drawing 2 , the design tool 1 has the Boolean part article tool information setting section (specification information generation section) 11 which generates the Boolean part article tool information 12 based on the Boolean part article information 4, and the main part 13 of a design tool (design section main part) which supports the design of the application by the combination of each Boolean part article based on the Boolean part article tool information 12 generated by the Boolean part article tool information setting section 11. In addition, the Boolean part article information 4 contains Boolean part article basic information 4a and Boolean part article related information (ID [of each Boolean part article], attribute information, I/O information, etc.) 4b (initial entry between each Boolean part articles) as definition information on the Boolean part article. Moreover, the Boolean part article tool information 12 expresses the specification information on each Boolean part article, and includes the I/O specification (Boolean part article I/O specification 12a) of each Boolean part article, the connection specification between each Boolean part article (Boolean part article connection specification 12b), and the attribute specification (Boolean part article attribute specification 12c) of each Boolean part article. In addition, the logical design information 5 outputted from the main part 13 of a design tool contains the attribute information on each Boolean part article (part attribute information 5a), and the initial entry between each Boolean part article (part initial-entry 5b).
[0026] Moreover, the design tool 1 has the screen input/output interface section 14 connected to the main part 13 of a design tool. Here, the screen input/output interface section 14 can offer the graphical user interface (GUI) which expresses as an icon two or more Boolean part articles it is unrefined on the basis of a design on a tool screen, and can express now the combination of the software parts which constitute the application used as the candidate for development by the graph structure which connected between icons with the related line. In addition, the screen input/output interface section 14 can be realized as an interface program which operates on applications, such as a web browser.
[0027] Drawing 3 is drawing showing an example of a tool screen. As shown in drawing 3 , the tool screen 20 has the pallet field 21 and the working area 22. Among these, in the pallet field 21, a list indication of two or more icons 23 corresponding to the Boolean part article it is unrefined on the basis of a design is given based on the Boolean part article information 4. Moreover, two or more icons 24 stuck based on the icon 23 arranged to the pallet field 21 are displayed on the working area 22. In addition, the icon 24 arranged at the working area 22 corresponds to each software parts which constitute the application used as the candidate for development.
[0028] In such a tool screen 20, a user can perform [input units, such as a mouse,] now various kinds of operations (attachment, movement, relating, etc. of an icon) to an icon, can be interlocked with the operation to these icons, and can design [the combination of the Boolean part article] application now.
[0029] After clicking the left button of a mouse on the arbitrary icons of two or more icons 23 specifically arranged to the pallet field 21 (icons A, B, and C), an icon 24 (icons A, B, and C) can be stuck on a working area 22 by clicking the left button of a mouse again in the arbitrary positions of a working area 22 (sign 26 reference). In addition, after doing in this way and sticking an icon 24 on a working area 22, the attribute of the Boolean part article etc. can be set up by clicking the right button of a mouse on an icon 24, carrying out pop-up one of the menu (not shown), and choosing the item in this menu suitably (names of the Boolean part article ("parent", "child1", etc.) etc.).
[0030] Moreover, the icon corresponding to the component of the origin which sets up a relation among the icons arranged at the working area 22 (in drawing 3) The icon corresponding to the component of the point which sets up a relation where it pushed the left button of a mouse on the output terminal of Icon A and a left button is pushed (in drawing 3) What a mouse is moved for to Icons B and C (it drags) can define the related line 25 between icons (sign 27 reference).
[0031] In addition, operations (arrangement of an icon, movement, relating, etc.) to the icon on the tool screen 20 are realized by basic operation, such as a mouse. In addition, basic operation, such as a mouse, is unified between the Boolean part articles.

[0032] Next, an operation of the gestalt of this operation which consists of such composition is explained.
[0033] First, drawing 4 and drawing 5 explain processing of the logic layer (design tool 1) of application development system shown in drawing 1 and drawing 2 .
[0034] Drawing 4 is a flow chart for explaining operation of the Boolean part article tool information setting section 11 of the design tool 1 shown in drawing 1 and drawing 2 .
[0035] As shown in drawing 4 , first, the Boolean part article tool information setting section 11 of the design tool 1 reads Boolean part article basic information 4a among the Boolean part article information 4 (Step 101), and generates Boolean part article I/O specification 12a and Boolean part article attribute specification 12c for every Boolean part article based on this Boolean part article basic information 4a (Steps 102 and 103). Next, Boolean part article related information 4b is read among the Boolean part article information 4 (Step 104), and Boolean part article connection specification 12b is generated for every combination of the Boolean part article based on this Boolean part article related information 4b (Step 105).
[0036] Drawing 7 (a) and (b) are drawings showing an example of the Boolean part article information 4. Among these, drawing 7 (a) shows the whole information containing a part of Boolean part article basic information 4a (ID of each Boolean part article), and Boolean part article related information 4b (initial entry between each Boolean part articles). Moreover, drawing 7 (b) shows the individual information containing a part of Boolean part article basic information 4a (each attribute information and I/O information on the Boolean part article).
[0037] Here, ID of each Boolean part article is described by the portion of the sign (A) of drawing 7 (a) (<component type="A"/> etc.). Moreover, the initial entry between each Boolean part articles it is unrefined the origin of Boolean part article connection specification 12b is described by the portion of a sign (B) (<connection from="A"to="B"/> etc.). It means that the portion of " [in addition,] <connection from="A"to="B"/>" is connectable with the software parts B from the software parts A.
[0038] Moreover, the attribute information on each Boolean part article it is unrefined the origin of Boolean part article attribute specification 12c is described by the portion of the sign (C) of drawing 7 (b) (<property name="name" type="string"/> etc.). It means that the portion of " [in addition,] <property name="name" type="string"/> <property name="option" type="string"/>" has the two character string attributes of "name" and "option". Moreover, the I/O information on each Boolean part article it is unrefined the origin of Boolean part article I/O specification 12a is described by the portion of a sign (D) (<terminalid="out" type="out" min=1 max=3> ... </terminal> etc.). in addition -- "-- < -- terminal id -- = -- "out" -- type -- = -- "out" -- min -- = -- one -- max -- = -- three -- > -- < -- connectwith type -- = -- " -- B -- " -- /-- > -- < -- connectwith type -- = -- " -- C -- " -- /-- > -- < -- /-- terminal -- > -- " -- a portion -- "out" -- ** -- saying -- an output terminal -- it is -- one -- a ** -- from -- three -- a **
[0039] Drawing 5 is a flow chart for explaining operation of the main part 13 of a design tool of the design tool 1 shown in drawing 1 and drawing 2 .
[0040] As shown in drawing 5 , the screen input/output interface section 14 of the design tool 1 presents the tool screen 20 shown in drawing 3 under control by the main part 13 of a design tool, receives operation of the icon by the user, and supports the design of the application by the combination of each Boolean part article (Steps 201 and 202).
[0041] Specifically, the screen input/output interface section 14 sticks an icon 24 (icons A, B, and C) on a working area 22 by clicking the left button of a mouse again in the arbitrary positions of a working area 22, after a user clicks the left button of a mouse on the arbitrary icons of two or more icons 23 arranged to the pallet field 21 of the tool screen 20 (icons A, B, and C) (Step 201).
[0042] Moreover, where it pushed the left button of a mouse on the output terminal of the icon (drawing 3 the icon A) corresponding to the component of the origin which sets up a relation among the icons arranged at the working area 22 and a left button is pushed, what a mouse is moved for to the icon (drawing 3 icons B and C) corresponding to the component of the point which sets up a relation (it drags) defines the related line 25 between icons (Step 202).
[0043] Then, when not appropriate, based on Boolean part article I/O specification 12a of the Boolean part article tool information 12, the validity of the node of each Boolean part article is checked (Step 203), and the main part 13 of a design tool returns to Step 202, continues processing, and when appropriate, it progresses to Step 205 (Step 204).
[0044] Moreover, by checking validity (Step 205), the main part 13 of a design tool returns the connection relation between each Boolean part article to Step 202, when not appropriate, based on Boolean part article connection specification 12b of the Boolean part article tool information 12, it continues processing, and when appropriate, it progresses to Step 207 (Step 206).
[0045] Then, a user clicks the right button of a mouse on the icon 24 stuck on the working area 22, and does pop-up one of the menu (not shown). If the item in this menu is chosen suitably and the attribute of the Boolean part article etc. is set up (Step 207) When not appropriate, based on Boolean part article attribute specification 12c of the Boolean part article tool information 12, the validity of the attribute of each Boolean part article is checked (Step 208), and it returns to Step 207 and processing is continued, and in being appropriate, it progresses to Step 210 (Step 209).
[0046] When it finally judges whether processing of all the Boolean part articles was completed (Step 210) and processing is completed, the logical design information 5 is outputted (Step 211).
[0047] Drawing 8 is drawing showing an example of the logical design information 5, and the logical design information 5 contains part attribute information 5a (refer to sign (E)) and part initial-entry 5b (refer to sign (F)). Here, the logical design information 5 shown in drawing 8 corresponds to the combination (Icon B (a name is "child1" and an option is "11") and Icon C (a name is "child2" and an option is "12") should be connected in the shape of a tree, and combine with the bottom of Icon A (a name is "parent" and an option is "1")) of the Boolean part article as shown in drawing 3 .
[0048] Next, drawing 6 explains the processing of the physical layer (the source generation processing section 2 and compiler 3) of application development system shown in drawing 1 .

[0049] Drawing 6 is a flow chart for explaining operation of the source generation processing section 2 shown in drawing 1 .
[0050] As shown in drawing 6 , the source generation processing section 2 generates a source code 7 based on the logical design information 5 (part attribute information 5a and part initial-entry 5b) and the physical part information 6 which were outputted by the design tool 1.
[0051] Drawing 9 and drawing 10 (a), (b), and (c) are drawings showing an example of the physical part information 6. Among these, drawing 9 shows the information relevant to the code generation method for every software parts among the physical part information 6, and contains the correspondence relation between the physical parts and the Boolean part articles which were mounted for [specific] plat forms, attribute related information, connection related information of each physical parts, etc. Moreover, drawing 10 (a), (b), and (c) show the information relevant to the whole code generation method.
[0052] Here, the initialization code generation method of each software parts is described by the portion of the sign (G) of drawing 9 (<new code="ClassX%name%=ClassX.newinstance()"/> etc.). Moreover, the attribute setting code generation method of each software parts is described by the portion of a sign (H) (<property name="option" code="%name%.addItem("OPTION", "%value%", "string")"/> etc.). Furthermore, the part connection setting code generation method of each software parts is described by the portion of a sign (I) (<code="%from%.addItem("CHILD", %to%);/> etc.). In addition, the portions of a sign (G) and (H) and the portion of a sign (I) are portions which become the origin of attribute setting code 7b of a source code 7, and part connection setting code 7c, respectively, and the applicable portion of a source code 7 is generated by replacing "%xxx%" of item with the value of the logical design information 5.
[0053] Moreover, the initial-setting code generation method which becomes the portion of the sign (J) of drawing 10 (a) the origin of initial-setting code 7a of a source code 7 is described (<init file="fw1init.java"/>), and the end processing setting code generation method which becomes the origin of end processing setting code 7d of a source code 7 is described by the portion of a sign (K) (<term file="fw1term.java"/>). in addition -- "-- < -- init file -- = -- " -- fw -- one -- init -- . -- java -- " -- /-- > -- " -- a portion -- drawing 10 -- (-- b --) -- being shown -- a template -- a file (fw1init.java) -- importing -- a sake -- description -- it is -- "-- < -- term file -- = -- " -- fw -- one -- term -- . -- java -- " -- /-- > -- " -- a portion -- drawing 10 --
[0054] Based on such physical part information 6, the source generation processing section 2 outputs initial-setting code 7a of a source code 7 first, as shown in drawing 6 (Step 301).
[0055] Next, part attribute information 5a is read among the logical design information 5 (Step 302), and attribute setting code 7b of a source code 7 is generated for every software parts based on this part attribute information 5a and the physical part information 6 (Step 303).
[0056] Then, when it returns to Step 302 and processing is continued, when it judges whether processing of all software parts was completed (Step 304) and processing is not completed, and processing is completed, it progresses to Step 305.
[0057] Next, part initial-entry 5b is read among the logical design information 5 (Step 305), and part connection setting code 7c of a source code 7 is generated for every software parts based on this part attribute information 5a and the physical part information 6 (Step 306).
[0058] Then, when it returns to Step 305 and processing is continued, when it judges whether processing of all software parts was completed (Step 307) and processing is not completed, and processing is completed, based on the physical part information 6, end processing setting code 7d of a source code 7 is outputted (Step 308), and the whole processing is ended. Drawing 11 is drawing showing an example of the source code 7 generated by doing in this way.
[0059] In addition, after it does in this way and a source code 7 is generated by the source generation processing section 2, as shown in drawing 1 , based on the mounting information on each physical part stored in the part library 8, the executable file 9 which can be performed on a specific plat form is generated by applying a source code 7 to a compiler 3.
[0060] Thus, the development of application which is performed with the combination of software parts according to the gestalt of this operation (1) Processing which designs application combining the Boolean part article which comes to extract the portion for which it does not depend on the system environment of a plat form etc. among software parts (logic layer), (2) Since it carries out by dividing into the processing (physical layer) which generates application combining the physical parts which are software parts depending on system environment, about each part of a logic layer Irrespective of final system environment, it can become possible to use in common and application corresponding to system environment can be developed by preparing each part of a physical layer individually according to final system environment. For this reason, application excellent in the maintainability which can respond to change of system environment flexibly can be developed easily.
[0061] Specifically based on the logical design information 5 acquired by processing of (1) logic layer (design tool 1), the application corresponding to two or more system environment is generable by substituting the parts of a physical layer, and processing (the source generation processing section 2, the physical part information 6, the part library 8, and compiler 3 grade). Moreover, the logical design information 5 acquired by processing of (2) logic layers can generate easily the application which can be performed on new system environment by preparing the parts of a physical layer and processing which were doubled with new system environment for every application, even when the system environment to perform changes a lot by technical progress etc., since it was rare to be influenced by system environment. Furthermore, since processing of (3) logic layers can be limited to outputting the specific logical design information 5, without being bound by the physical layer, it can prepare easily the development tool which operates on two or more system environment.
[0062] Since the logical design information 5 which connects processing of a logic layer and processing of a physical layer is described in the high language of the versatility of an XML language according to the gestalt of this operation, flexibility can be given to the both sides of a physical layer and a logic layer, substitution of each layer can be made easy, and, moreover, the life of the logical expression of operating processing of application can be lengthened.

[0063] In addition, while the source generation processing section 2 generates the source code 7 containing an attribute setting code and a part connection setting code in the gestalt of operation mentioned above Although it is made to generate the executable file 9 to which the attribute of software parts and the connection relation between software parts were set by the compiler 3 based on this source code 7 You may make it set up about the attribute of software parts, and the connection relation between software parts at the time of execution like the gestalt of other operations shown in drawing 12 or drawing 17 .
[0064] Hereafter, drawing 12 or drawing 17 explains the gestalt of other operations of the application development system by this invention. in addition, the gestalt of the operation which shows others to drawing 1 or drawing 11 except for the point that the gestalt of operation shown in drawing 12 or drawing 17 set up the attribute of software parts, and the initial entry between software parts at the time of execution of an executable file and abbreviation -- it is the same In the gestalt of operation shown in drawing 12 or drawing 17 , the same sign is given to the same portion as the gestalt of operation shown in drawing 1 or drawing 11 , and detailed explanation is omitted.
[0065] As shown in drawing 12 , in the gestalt of other operations of this invention, the attribute of software parts and the initial entry between software parts are set up by the executive operation section 15 and the part object generation processing section 16 at the time of execution of the executable file 9 which generated the source code 7 containing an interpreter die-parts object resultant code by source generation processing section 2', and was generated by the compiler 3 based on this source code 7.
[0066] Drawing 13 is a flow chart for explaining operation of source generation processing section 2' shown in drawing 12 .
[0067] As shown in drawing 13 , source generation processing section 2' generates a source code 7 based on the physical part information 6.
[0068] Drawing 15 and drawing 16 (a), (b), (c), and (d) are drawings showing an example of the physical part information 6. Among these, drawing 15 shows the information relevant to the part object generation method for every software parts among the physical part information 6. Moreover, drawing 16 (a), (b), (c), and (d) show the information relevant to the whole code generation method.
[0069] The initial-setting code generation method which becomes the portion of the sign (L) of drawing 16 (a) the origin of initial-setting code 7a (refer to drawing 13) of a source code 7 is described (<init file="fw1init.java"/>). The part object generation method which becomes the portion of a sign (M) the origin of interpreter die-parts object resultant-code 17e (refer to drawing 13) of a source code 7 is described (<main file="fw1main.java"/>). The end processing setting code generation method which becomes the origin of end processing setting code 7d (refer to drawing 13) of a source code 7 is described by the portion of a sign (N) (<term file="fw1term.java"/>). The portion of " [in addition,] <init file="fw1init.java"/>" It is description for importing the template file (fw1init.java) shown in drawing 16 (b). The portion of " <main file="fw1main.java"/>" It is description for importing the template file (fw1main.java) shown in drawing 16 (c). The portion of " <term file="fw1term.java"/>" is description for importing the template file (fw1term.java) shown in drawing 16 (d).
[0070] Based on such physical part information 6, source generation processing section 2' outputs initial-setting code 7a of a source code 7 first, as shown in drawing 13 (Step 401).
[0071] Next, interpreter die-parts object resultant-code 7e of a source code 7 is outputted (Step 402).
[0072] Then, based on the physical part information 6, end processing setting code 7d of a source code 7 is outputted (Step 403), and the whole processing is ended. Drawing 17 is drawing showing an example of the source code 7 generated by doing in this way.
[0073] In addition, after it does in this way and a source code 7 is generated by source generation processing section 2', as shown in drawing 12 , based on the mounting information on each physical part stored in the part library 8, the executable file 9 which can be performed on a specific plat form is generated by applying a source code 7 to a compiler 3.
[0074] Thus, although the generated executable file 9 is performed by the executive operation section 15, the executive operation section 15 calls the part object generation processing section 16 in process of the processing, and thereby, it sets up the attribute of the software parts concerned, and the initial entry between software parts while it generates the part object of software parts.
[0075] Drawing 14 is a flow chart for explaining operation of the part object generation processing section 16 shown in drawing 12 .
[0076] As shown in drawing 14 , the part object generation processing section 16 performs a setup of the initial entry between generation of a part object, an attribute, and parts etc. based on the logical design information 5 (part attribute information 5a and part initial-entry 5b) and the physical part information 6 which were outputted by the design tool 1.
[0077] Specifically, part attribute information 5a is read among the logical design information 5 (Step 501), a part object is first, generated for every software parts based on this part attribute information 5a and the physical part information 6, and an attribute is set up (Step 502).
[0078] Then, when it returns to Step 501 and processing is continued, when it judges whether processing of all software parts was completed (Step 503) and processing is not completed, and processing is completed, it progresses to Step 504.
[0079] Next, part initial-entry 5b is read among the logical design information 5 (Step 504), cross-reference information is set up for every software parts based on this part attribute information 5a and the physical part information 6, and between parts is connected (Step 505).
[0080] Then, the whole processing is ended when it returns to Step 504 and processing is continued, when it judges whether processing of all software parts was completed (Step 506) and processing is not completed, and processing is completed.
[0081] In addition, in the gestalt of operation shown in the gestalt and drawing 12 , or drawing 17 of the operation shown in drawing 1 or drawing 11 mentioned above, although the case where application development system was realized on a single computer was mentioned as the example and explained, it is possible not only this but to realize by distributed development

environment as shows application development system to drawing 18 . A logic layer (the design tool 1 and the Boolean part article information 4) is prepared on a client computer 31, and you may make it specifically, prepare a physical layer (the source generation processing section 2, a compiler 3, the physical part information 6, a source code 7, the part library 8, and executable file 9) on the server computer 33 in two or more client computers 31 and server computers 33 which were connected through the networks 32, such as the Internet, as shown in drawing 18 . In addition, since the logical design information 5 by the XML language without the mounting information on physical parts etc. will be delivered between a client computer 31 and the server computer 33 in this case, traffic can be stopped. For this reason, it becomes possible also in the Internet environment to develop the application by the side of [a client computer 31 side to] the server computer 33, and it becomes possible to apply to service which offers the open application development environment in the Internet environment.

[0082] Moreover, although the source code 7 and the part library 8 which were generated by the source generation processing section 2 with the compiler 3 are unified, you may make it unify not only this but the source code 7, and the part library 8 by the part selection technique, the part plug-in technique, etc. in the gestalt of operation shown in the gestalt and drawing 12 , or drawing 17 of the operation shown in drawing 1 or drawing 11 mentioned above.

[0083] Furthermore, the gestalt of operation shown in the gestalt and drawing 12 , or drawing 17 of the operation shown in drawing 1 or drawing 11 mentioned above can be described in arbitrary form, such as not only this but a tabular format, although the Boolean part article information 4 and the physical part information 6 are described with the XML language.

[0084] In addition, in the gestalt of operation shown in the gestalt and drawing 12 , or drawing 17 of the operation shown in drawing 1 or drawing 11 mentioned above, the design tool 1, the source generation processing section 2, and the compiler 3 of application development system are realizable as a program. If the application development program which consists of such a program is distributed development environment as shown in drawing 18 , it can be recorded on the record medium which a flexible disk 34 and CD-ROM35 grade can computer read, and can develop application according to a procedure as shown in a procedure as shown on a client computer 31 and the server computer 33 at drawing 4 or drawing 6 , drawing 13 , and drawing 14 .

[0085] In addition, as a record medium in this invention, like a magnetic disk, an internal memory, a hard disk, optical disks (CD-R, DVD (Digital Versatile Disk), etc.), magneto-optic disks (MO (Magneto Optical) etc.), and semiconductor memory, it may be restricted to neither a flexible disk nor CD-ROM, but a program can be recorded, and as long as it is the record medium which a computer can read, the record form may be one of form. Moreover, as a record medium, the subcarrier and communication-of-information medium at the time of being transmitted on a network are also included.

[0086] Moreover, you may perform a part of each processing for MW(s) (middleware), such as OS (operating system) which is working on a computer based on directions of the program installed in the computer from the record medium, and database management software, network software, etc. realizing this invention.

[0087] Furthermore, the record medium which the record medium in this invention downloaded the program transmitted by not only the medium that became independent of a computer but LAN, the Internet, etc., and was memorized or stored temporarily is also contained.

[0088] A record medium may be contained in the record medium in this invention further again, not only one but when processing in this invention is performed from two or more media, and medium composition may be which composition.

[0089]

[Effect of the Invention] As explained above, according to this invention, application excellent in the maintainability which can respond to change of system environment flexibly can be developed easily.

[Translation done.]